In the Claims

The following listing of the claims replaces all previous listings.

1. (Currently Amended) A method for preventing overrun of an input data buffer within a program having the input data buffer on a stack data structure, the program executing on a computing system, the method comprising:

pushing all arguments to a function onto the stack data structure;

pushing a return address onto the stack data structure for use in obtaining a memory address for an instruction to be executed upon completion of the function;

pushing onto the stack data structure a security token, the security token comprising a randomly generated data value determined using a random number generator once each time the program is executed, wherein the random number generator generates the randomly generated data value using a snapshot of a system clock within the computing system obtained before the program first accepts input data;

allocating memory locations on the stack data structure for use as local variables within the function;

completing instructions within the function;

retrieving the security token value from the stack data structure; and

if the retrieved security token value is identical to the randomly generated data value, returning from the function using the return address stored on the stack data structure.

2. (Previously Presented) The method according to claim 1, wherein the method further comprises aborting operation of the program if the retrieved security token value is not identical to the randomly generated data value.

3. (Canceled)

4. (Canceled)

5. (Original) The method according to claim 1, wherein the function comprises a subroutine that does not return a data value.

6. (Currently Amended) The method according to claim 1, wherein the function comprises a subroutine that does ~~returns~~ return one or more data values.

7. (Currently Amended) An apparatus for preventing overrun of an input data buffer within a program having the input data buffer on a stack data structure, the program executing on a computing system, the apparatus comprising:

a function call module placing arguments to a function and a return address onto the stack data structure;

a push security token module placing onto the stack data structure a security token, the security token comprising a randomly generated data value determined using a random number generator once each time the program is executed, wherein the random number generator generates the randomly generated data value using a snapshot of a system clock within the computing system obtained before the program first accepts input data;

a perform function module performing operations within the function, the perform function module allocating memory locations on the stack data structure for use as the input data buffer;

a pop security token module retrieving the security token from the stack data structure upon completion of operation of the perform function module;

a test module comparing the retrieved security token with the randomly generated data value; and

a complete function module completing operations of the function;

wherein the complete function module returns from the function if the retrieved security token is determined to be identical to the randomly generated data value by the test module.

8. (Previously Presented) The apparatus according to claim 7, wherein the complete function module aborts operation of the program if the retrieved security token is determined not to be identical to the randomly generated data value by the test module.

9. (Canceled)

10. (Canceled)

11. (Currently Amended) The apparatus according to claim [[9]] 7, wherein the function comprises a subroutine that does not return a data value.

12. (Currently Amended) The apparatus according to claim [[9]] 7, wherein the function comprises a subroutine that does ~~returns~~ return one or more data values.

13. (Currently Amended) A computer program product readable by a computing system and encoding a set of computer instructions implementing a method for preventing overrun of an input data buffer within a program having the input data buffer on a stack data structure, the program executing on a computing system, the method comprising:

pushing a return address onto the stack data structure for use in obtaining a memory address for an instruction to be executed upon completion of a function;

pushing onto the stack data structure a security token, the security token comprising a randomly generated data value determined using a random number generator once each time the program is executed, wherein the random number generator generates the randomly generated data value using a snapshot of a system clock within the computing system obtained before the program first accepts input data;

completing the instructions within the function;

retrieving the security token value from the stack data structure;

if the retrieved security token value is identical to the randomly generated data value, returning from the function using the return address stored on the stack data structure.

14. (Previously Presented) The computer program product according to claim 13, wherein the method further comprises aborting operation of the program if the retrieved security token value is not identical to the randomly generated data value.

15. (Canceled)

16. (Canceled)

17. (Previously Presented) The computer program product according to claim 13, wherein the function comprises a subroutine that does not return a data value.

18. (Currently Amended) The computer program product according to claim 13, wherein the function comprises a subroutine that does ~~returns~~ return one or more data values.

19. (Previously Presented) The computer program product according to claim 13, wherein the computer data product comprises a set of computer instructions encoded and stored onto a computer-readable storage medium.

20. (Previously Presented) The computer program product according to claim 13, wherein the computer data product comprises a set of computer instructions encoded within a carrier wave for transmission between computing systems.